

Tests of Dedicated Panda Queues, Condor Flocking, and FAX Access

Frederick Luehring
luehring@indiana.edu

Indiana University

11-Dec-2013

US ATLAS Distributed Computing Workshop

Introduction

- Three ways of providing physics analysis computing using remote resources have been compared to the well-loved Local Tier 3 Centers (LT3C):
 1. Dedicated Panda Queues ultimately with user pilots
 2. Condor flocking
 3. Federating ATLAS storage systems using XRootD (FAX)
- This work is driven by paucity of funding for new hardware to replace the aging LT3C hardware.
 - Much of the hardware was purchased with stimulus (ARRA) funding in 2010 and is approaching end of life.
- Studying cloud computing and virtualization technology is NOT included in this work.

Test Scenario

- The test scenario (use case) is based on the typical code development/cut tuning work flow of repeatedly running short tests of a series of small changes.
 - That cycle is locally edit the code or scripts, locally compile the code, run a test on a small dataset, look at the results and repeat a large number of times.
 - The tests of this study were aimed at minimizing the time spent running of the test and getting the results back to the user's site.
 - For comparison, a baseline of doing this cycle with code running on a Tier 3 with local input files was established.
- All of the methods of using remote resources mentioned on the previous page were tested using this scenario.

Test Job

- The test job used an example package making a cut flow skimming analysis of Standard Model WZ D3PDs developed by Doug Benjamin.
 - The SMWZ D3PD example reads a large input file containing D3PDs and writes a small output D3PD file containing the events that pass the cuts.
- Please note that this example job results in an I/O bound situation with a small CPU usage per event.
 - Jason Nielson pointed out to me that a work flow with a much higher compute time per event can take too long to be practical to run locally even if the dataset is only a few TB.
- So local running is most suitable for job flows where the combination of dataset size and time per event results in a turn around time of overnight or less.

Accessing the ATLAS Software

- The software environment was setup using the ATLAS Local Root Base system which takes the software from the ATLAS offline cvmfs repository.
 - The commands to setup the needed software were setup using the setupATLAS script.
 - The asetup command and pathena were not used.
 - Among ATLAS physics analysis community, this is by far the most common way to access the offline software.

Local Running Baseline

- Interactive running on the IU Tier 3 established a baseline for how quickly a short test could be done.
 - The results were highly dependent on whether the input data was cached in memory which happens the first time the program is run with succeeding jobs taking only ~40% of initial run time once the data was cached.
 - How long the data stays cached depends on the amount of I/O activity on the cluster and the amount of memory available for caching.
 - To put some numbers on the data processing speed, the SMWZ D3PD example running on Dell Servers processed in 2010 could process 5 input files that were each ~4-4.5 GB in ~0.7 minutes (cached) or ~1.9 minutes (cached).
 - NB: The example only reads the data needed by the SMWZ cut flow and thus only about 50-70 MB of the 4 GB in each input data file was read (varies some depending on dataset).

Longer Local Running Tests

- Initially longer interactive tests were run using larger numbers of input files (up to ~25). The processing time per file was about similar to the shorter test.
 - When the input data was cached in memory, similar improvements to the 5 input file tests were observed.
 - Given how little data is actually used, even when using many input files, the needed data was usually cached.
 - When a job runs at a remote site, the data will not be cached and the longer time associated with uncached data is typically observed.
 - Scaling up to a huge dataset as was possible using FAX, leads to a test where caching is not important.

Tests With A Local Batch System

- Tests were also run using the local batch system.
 - The batch system the Grid Engine batch system backed by 9 Dell R610 servers with 8 physical cores (16 cores with hyper-Threading) and 4 GB memory per physical core.
 - In the CPU chips used in these servers, an improvement of about 1/3 and not a doubling of throughput is obtainable using Hyper-Threading.
 - The tests used a 110 file 430 GB dataset stored on a local storage system accessible via a 10 Gbps local network.
 - The results varied between those observed previously with cached and uncached data with a small additional overhead added by the batch system.
- It was possible to easily abort the job sets with a single command. This allowed type of flexibility that makes a LT3C desirable to the analysis community.

Tests with a Dedicated Panda Queue

- Testing the moved to non-local resources initially using the prun client to submit job sets from the Indiana University (IU) Tier 3 to Panda queues.
 - Most of the initial work used a dedicated panda queue of 40 slots that sit idle wait for test jobs was established at AGT2 (with help from Bob Ball).
 - A dataset of 4 TB dataset with 1113 SMWZ D3PD files was used for this testing but in the end, nearly all job sets were 40 jobs each with 5 input files totaling ~20 GB which is the usual limit for the aggregate size of the input files at most sites. This allowed filling each dedicated slot with one job.
 - A few job sets were also run in the normal MWT2 and AGLT2 analysis queues to gauge what sort of turn around a light user of the shared resources might experience.

Test with Shared Panda Queues

- The set of 40 SMWZ D3PD example prun jobs were submitted to the normal AGLT2 & MWT2 analysis queues where they competed with other analysis jobs for job slots and job turn around was observed.
 - Most jobs returned within an hour or two which is fast considering the shared nature of the queue and the often staggering number jobs waiting to run.
 - I did not run enough job sets to seriously reduce my priority.
 - While very good turn around times were achieved for submission to remote sites, the turn-around was variable and understandably far slower than the turn around on a local resource.
 - On the plus side prun and Panda took care of almost all the job splitting and data handling issues. Retries of failed jobs were also done automatically.

Dedicated Panda Queue Tests

- The test jobs were sent to a dedicated test queue of 40 cores at AGLT2 (slots wait empty for jobs).
 - Turn around times of ~30 minutes were observed for payload jobs that ran in ~2-3 minutes (usual 20 GB input).
 - When AGLT2 was busy, a few minutes were added to the turn around time.
 - One could start downloading the output files with dq2-get almost as soon as the Panda monitor or pbook showed the run jobs as finished (well before the official job end).
 - pbook returns job status information quicker than the Panda Monitor
 - Sending the output files to a nonvolatile storage element using the --destSE option worked well to preserve the output but typically took several hours to complete.
 - --destSE is not useful to return job output to a waiting interactive user.
 - It is hoped that RUCIO local caches will allow rapid, automatic return of the job set output in the future.
 - Aborting a job set took several minutes (though less than the 30 minute time suggested when aborting a job set).

Dedicated Panda Queue & No Build

- Experimentation led to the correct prun syntax to completely skip the prun build job and send the executable compiled locally on the Tier 3.
 - This saves ~10 minutes leading to ~20 minute turn-around.
 - Even on a dedicated queue there was job to job turn around time variation and some jobs ran longer.
 - The 20 minute turn around was measured by the time stamps in the end of job email and did not include ~1 minute of that prun takes to launch the job.
 - There are two similarly named prun options about skipping the build but to entirely skip the build job use --noBuild.
- In the repetitive code development use case being tested, the developer will always builds locally and then runs a short test before submitting to the grid so sending the executable is basically free.

Ded. Queue, No Build, User Pilots

- With help from Jose Caballero and John Hover, a wrapper script was used to prime the dedicated slots with pilots waiting for the jobs to come.
 - Hiro Ito solved a privilege issue by providing a user area in the file system used by the LFC to register job set output. This allowed the wrapper script pilots to succeed using the non-production usatlas3 role assigned to my grid certificate.
 - Paul Nilsson added an option (-G) to the pilot to allow the user to set the pilot lifetime a user defined value longer than the usual 3 minutes. Thus pilots can wait hours checking once a minute for a job and run many jobs over their lifetime.
 - The wrappers were sent directly to the test queue using a condor schedd running at the IU Tier 3.
 - NB: The wrapper script needed a long lived (> 1 day) x509 proxy and I believe had to be the same as one used by prun.

Ded. Queue, No Build, User Pilots

- The results are encouraging: job sets still take 20 minutes to officially complete but usually all of the member jobs are done in less than 5-10 minutes.
 - One can start downloading soon after the individual run jobs complete well before Panda marks the job set as done.
 - For reasons that are unclear to me, recently there have been cases where data access fails for one or more jobs and there is a 10 minute delay before the job is automatically retried.
 - The delay is painful for this code development work flow but no user intervention is required to restart the job.
 - There does also seem to be a 2-3 minute variation in when the jobs get access to the input data files. This is presumably dependent on the overall load at AGLT2.
- One additional thing to try is running a private AutoPyFactory but I haven't had the time to do this.

Condor Flocking Setup

- Since the IU Tier 3 uses grid engine and not condor as the batch system, it was not possible to flock jobs directly.
- A standalone local submit host running the condor schedd daemon was used to flock jobs to a Bosco server running at MWT2.
 - In the end thousands of jobs were successfully sent via Bosco showing that flocking jobs from a non-condor queue works well.
 - This is critical in the a case of shared resource where the analysis group can't change batch system to condor.
 - The initial site name based job authentication system was changed to a certificate based authentication system.
 - A tiny amount of pain produced a huge improvement in security against unauthorized use of the MWT2 condor queue.
 - Dave Lesny's help was essential in getting started.

Small Scale Condor Flocking Tests

- A test job was created that did the same analysis as the first job of the 40 job set used to test the dedicated Panda queue at AGLT2 by doing this:
 - The input dataset used by the AGLT2 tests was replicated to the MWT2 local group area.
 - A condor submission file was created that uploaded three files needed to run the job at MWT2:
 - A tarball was created from the directory tree of the built SMWZ D3PD example code at IU Tier 3.
 - A text file containing the URIs of the input datafiles.
 - A file containing the X.509 voms proxy.
 - The submission also sent a small shell script to execute the job. Two arguments were given to the shell script the name of the input text file containing the URIs and the name of the output file containing the skimmed events.

Small Scale Condor Flocking Tests

- The shell script was quite simple in what it did:
 - It setup the software environment using the setupATLAS script of the ATLAS Local Root Base system to access the software from cvmfs.
 - It used localSetupEmi to setup the EMI middleware to use the proxy file sent with the job to allow grid access to the input datafiles.
 - It used localSetupROOT to setup ROOT to allow the SMWZ D3PD example executable to find the needed runtime libraries.
 - It unpacked the tarball containing the example code tree
 - It then invoked the SMWZ D3PD example using the specified input file of URIs and writing to the output file.
- The submission file specified that the (small) output file was returned to the directory from where the job was submitted. The submission file also specified that the files containing sysout and syserr were returned to a subdirectory of the submission area. A condor-generated log file showing the condor messages about the job was also returned.
- This test is ideal for condor as it generates a small output root file. Bigger outputs would have filled the disk.

Small Scale Condor Flocking Results

- The tests were run with standard condor commands:
 - condor_submit was used to enter the job and the schedd daemon routed the resulting job to the MWT2 Bosco server.
 - condor_q command was used to examine the status of jobs as they progressed through processing.
 - condor_rm was used to abort jobs (or job sets in larger scale tests) and could quickly clear the queue of pending and running jobs giving good flexibility to clear the system and start another job (or job set).
- The results were good: generally the test job returned in about 3 minutes if slots were available – the same amount of time as the execution time of the job in Panda without the other overheads.
 - At times the test jobs waited hours for a job slot as happens at any shared batch system.

Scaling Up to Large Condor Job Sets

- It was easy to increase by brute force the number of jobs created by adding additional job submission lines after specifying new input text files containing different input file URIs.
 - However it wasn't very nice. I could have written a script to add the lines to the end of the submission but the command to submit a job can simultaneously submit any number jobs specified by the user and I wanted to use this facility
 - Also each submission line resulted in a new job set id number making it harder to tell which jobs were part of a set.

Scaling Up to Large Condor Job Sets

- A carefully constructed submission file was written that submitted all jobs in a single, large job set specifying the number of jobs in a single submission file command: `Queue <njobs>`
 - Getting the output log files numbered with leading zeros so that they appeared in correct numeric order when using `ls` required help from a condor developer who provided a an amazing uber-geek line for the job submission file.
 - The shell script used to execute the jobs also became more complex because it now had to be aware of which job in the job set it was running and act that information.
 - The script had to handle putting the same numbering with leading zeros on the output root file from each job in the job set.
 - The execution script was also modified to select 10 URIs based on the job number from single input file containing all URIs in the input dataset avoiding the need to creating hundreds of files containing 10 URIs each. Idea of Ilija Vukotic.

Scaling Up to Large Job

- A shell script was written to submit each job set from its own working directory containing all input files (except data files). The output ROOT & log files and condor log file were returned to this directory.
- The submit and execute scripts could also submit job sets to the local batch system using either local files or remote files accessed via FAX.
- The wrapper script took as its arguments:
 - The working directory
 - The number of jobs
 - The name of the input text file containing the URIs
 - A number indicating job type: local with local files, local with remote files via FAX, or condor flocking to MWT2.

Large Scale Condor Test Results

- The large scale test used the SMWZ example with 3169 input files total size 12 TB. The submission script create 317 jobs each with 10 input files with total size 40 GB.
 - NB: Recall that ~50 MB is read from each 4 GB input file.
- When slots were available, turn around time was quite impressive with all 317 jobs being completed within an hour.
- When MWT2 was busy it could take several hours for the first job to run and 2-3 more hours for all jobs to be completed.
- On a few occasions job sets stalled because of storage problems or FAX issues.

URI Specification

- Overtime the URI format used by both condor and FAX access changed several times.
 - The URI format was taken from the Panda Monitor info about test jobs and later Ilija Vukotic supplied the FAX URIs.
 - During the course of the testing, the MWT2 storage system had several dCache related upgrades and was converted to the RUCIO naming convention.
- Enabling the storage servers to failover proved essential for reliable operation using FAX URIs.
 - `export XrdSecGSISRVNAMES="*"`
- The RUCIO URIs are a lot shorter and according to Ilija about 10 times faster to look up.
 - For now it seems the RUCIO URIs are the clear choice.

Comparison

- Both prun and condor work and are pretty reliable.
 - Both can send the locally compiled executable with the job.
- Panda (prun) excels at doing the overhead of dealing with the input and output files using the DDM system.
 - Condor flocking requires the user to devise their own method of splitting up the input dataset into run jobs though the user may have done much of what is needed to use local queues.
 - It is up to the user to get condor to generate a sensible naming scheme for the output. Using the most obvious scheme, fixed names, can overwrite output files from previous jobs if the user is not careful.
- The Panda Monitor is really nice for tracking progress and finding results later).
 - Condor does not have a similar facility.

Comparison

- A manual dq2-get operation is needed to return the output files of a prun/pathena to the user's site while for small files condor returns them automatically to the submission directory.
 - This dq2-get can be started before the prun job is officially over (user receives email).
- If the output files are too big condor to return to the submission area, the job execution script needs to upload the files to DDM.
 - There is a system called “FAX Box” available at MWT2 could help here but I haven't had time to study it.
 - It should be possible to use dq2-put in the execution script to upload output files to DDM but I haven't tried it.

Comparison

- There are overheads before/after Panda runs the jobs that lengthen the turn around by ~15 minutes.
 - Even with user pilots waiting for for the jobs, the minimum time to fully complete a prun jobs is ~20 minutes though one can start transferring the job output before the job set officially ends.
 - Possibly a user written AutoPyFactory could help here.
- Panda will retry jobs automatically if they fail because of data access issues. Usually the job succeeds on the second attempt.
 - It is up to the user to resubmit failed jobs in condor
 - This also leads cases where just one or two jobs add 10 minutes to the total turn around time.

Comparison

- The pathena/prun --help options provides a daunting amount of output but the TWiki has good examples.
 - There is lots ATLAS specific information available.
- The condor clients provide basic information in response to any unknown option.
 - The condor team has provided a good set of basic submission file examples but it is hard to find examples for advanced usage.
 - The man pages for the condor commands are very detailed.
 - There don't seem to be examples for some of the condor commands. For example I wanted to use condor_qsub since the local Grid Engine batch system uses qsub. However my condor_qsub command failed for unspecified file not being found. I tried multiple web searches without ever finding a single example of using condor_qsub. There may be examples but search strings don't find them.

Take Away

- Panda handles a lot of boring details for you but has overheads slow throughput (most noticeable for short jobs). It's a heavier solution.
- Condor is fast and lightweight but you do a lot more bookkeeping and you have to do something to deal with large output datasets. It's lighter solution.
- What a user chooses depends on what they are doing. Bigger, longer running jobs would favor panda while small, short, repetitive tests would favor condor.
- To me the jury is still out and even after a lot of work, both approaches can be further optimized and tuned.
- One thing is true: having some dedicated resources are better than having only shared resources.

User Experience

- A lot of the user experience depends on the details of how the system prioritizes the jobs.
 - In my simple I/O limited tests with dedicated resources (panda at AGLT2) or very high priorities (condor at MWT2), the round trip time was usually really good but in the real world with jobs that are significantly more CPU intensive and need to be run many times this is not guaranteed.
 - At some point a user runs into a fair share mechanism.
- The team that setup the condor queue was quite concerned about the amount of their time that it would take to setup, maintain, and deal with users in the condor flocking.
- Panda has lots of support in place. There is a large funded effort to upgrade the production system.

Intro to FAX Studies

- FAX allows local jobs to read data from remote sites.
 - This is the exact opposite of the normal LHC mantra of “send the data to the jobs”. With FAX the data comes to the jobs “on the fly” essentially transparently when everything works.
 - Any modern OS will cache data in memory the first time it is read so later access to that data will be from local memory instead of the remote site making testing FAX access with short interactive jobs difficult.
 - The same effect was observed in local running.
 - Doug Benjamin has pointed out to me that adding a local caching mechanism for FAX would be desirable.
 - A good example of the effectiveness of local caching is the way cvmfs uses squid to locally cache only the ATLAS software versions being used at site. These files used over and over without putting additional load on the network. It all works, as long as the partition used for caching is big enough.

FAX Networking Considerations

- Like any system accessing data from the WAN, the quality and speed of the network matters.
 - FAX will work on a 1 Gbps WAN connection but for the best user experience, a 10 Gbps WAN connection is needed.
 - It is better for the Tier 3 users and lots better for everyone else in the building if the network connection is on a segregated research network and not on the commodity network used for email, web access, etc.
 - Having the Tier 3 in (correct term: advertised to) LHCONE uses a layer 2 protocol to route data to/from LHC sites.
 - Setting MTU to 9000 (“jumbo frames”) improved transfer rates.
 - Having the worker nodes directly connected to the WAN avoids the head node becoming a bottle neck.
- The IU Tier 3 used to test FAX WAN data access is on a research network, has a 10 Gbps connection, is advertised to LHCONE, uses jumbo frames, but the worker nodes are on a NAT behind a head node.

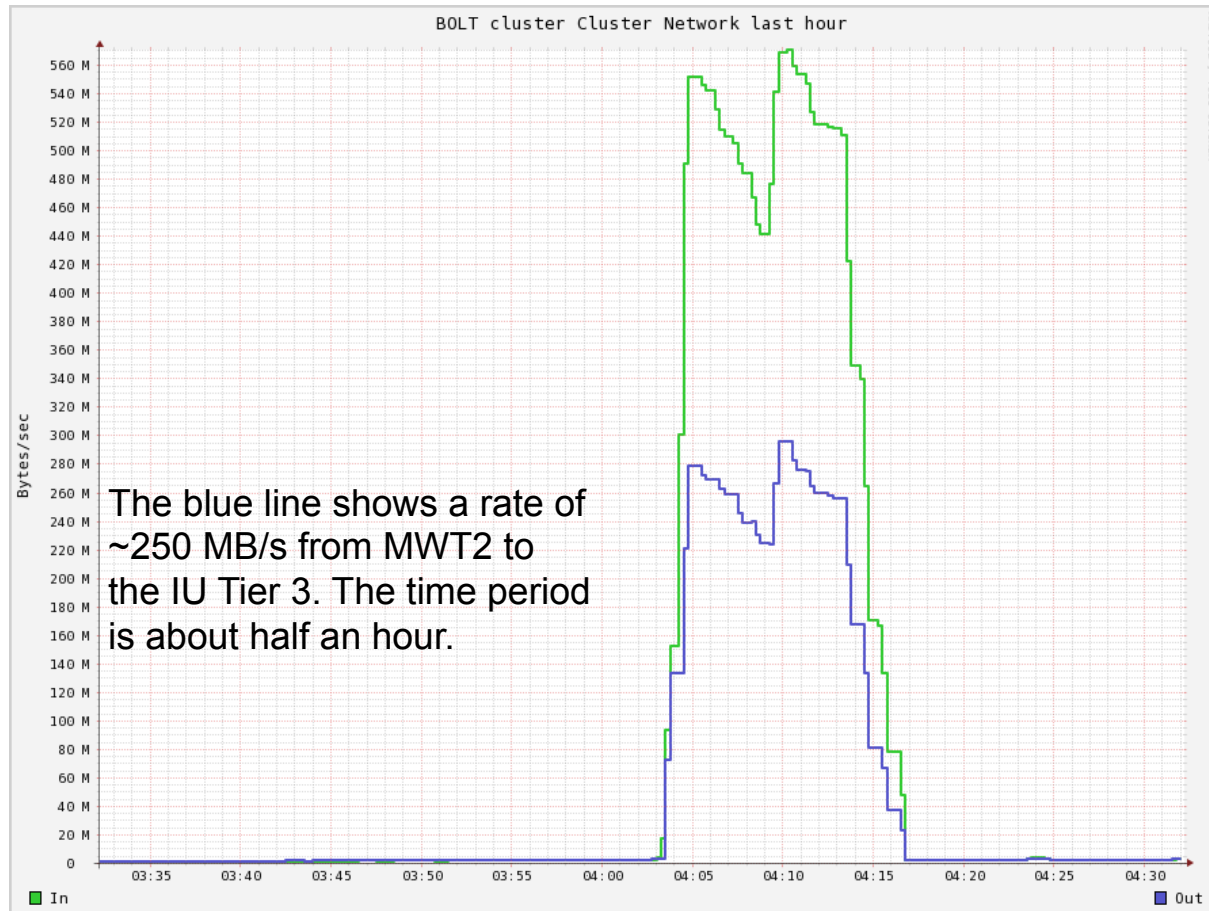
Small Scale Fax Test

- After consultation with Ilija Vukotic, interactive tests were run on IU Tier 3 using the local installation of the SMWZ D3PD Example code.
- The example code was not recompiled. Only the URIs in the text file specifying the input were changed to FAX URIs pointing at MWT2.
- The interactive tests ran without problems on all of the URIs and the running time was short: 2-3 minutes on the standard 20 GB set of input files.
 - As in regular running, the amount of data was small ~50 MB per 4 GB input file.
 - It was hard to measure the affect of the remote connection because just as with the local files, after the first test run, the data was cached in memory and giving great performance.

Large Scale FAX Tests

- The same SMWZ D3PD example 317 job set used to test condor was submitted locally to the IU Tier 2 with input URIs specifying FAX access to MWT2.
 - Originally all 144 slots were used but after an incident where all running jobs hung blocking other users from running, the test was restricted to 72 slots.
 - The large amount of data read was too big to be cached allowing a good measurement of FAX access rates.
- The results were good, it possible to get the entire set jobs done in less than an hour with a sustained data flow from MWT2 to the IU Tier 3 of ~250 MB/s.
 - This was true using both 144 and 72 job slots.
- However there were occasional problems that caused some tests to go more slowly or hang.

FAX Throughput of a Successful Job



The green line is twice the data rate from MWT2 because the plot shows the aggregate rate data rate for the IU Tier 3 and internal transfers and external transfers are combined on the plot.

Reasons for Problems

- There were lots of upgrading going on:
 - A number of upgrades were made to the underlying dCache storage system. There was also the RUCIO renaming. It took time to debug the changes.
 - There were also changes to the network for 100 Gbps connectivity upgrades.
 - There were al changes to the FAX name to name server.
 - Ilija pointed out that the SMWZ example opens all of files during initialization resulting in huge load spike. The code does this to be sure that the input files are available.
 - Much of the time the backend handled this fine with as many 144 jobs starting at the same time. After a short pause, the jobs soon progressed. However sometimes this seemed to deadlock
- Things to have stabilized now and no problems have been seen in several weeks.

Where FAX is not Suitable

- If a job is more limited by CPU than I/O and the dataset is big (~ TBs), even if FAX reads the data quickly from the remote location, processing the data at the Tier 3 may be too slow for lack of local CPU.
 - During the next running period the dataset will quadruple in size and the number cores at Tier 3 will probably stay the same or decline.
- So FAX is good for looking at small to medium size datasets and rapid testing/prototyping but it currently would not be sensible to process a huge remote dataset running on a Tier 3. Clearly this scenario is more suitable for the Panda or condor approach.

FAX Final Thoughts

- For the right kind of task, a Tier 3 with a good network connection can make great use of FAX.
 - A job can get access to almost any dataset and still run locally.
 - ROOT minimizes the amount data that actually gets transferred so if a small fraction of the information in the dataset is needed, local jobs run quickly.
 - Adding a local caching system would improve things further.
- These results suggest that it is desirable to have a global plan to improve network connectivity at the US Tier 3s.
- There have been problems but things are stable now.
- These tests only concern running local jobs on remote data. Clearly there are many other good ways that FAX can be used.

What To Do Next

- Test running an user AutoPyFactory for the AGLT2 dedicated test queue to reduce Panda turn around.
- Look into ways to help user with condor flocking.
- Study the use of FAX Box for handling large output files when using condor flocking.
- Submit more condor and FAX jobs to get a better feel for turn around times and rarer failure modes.
- Setup a condor batch system at the IU Tier 3 to study native condor flocking.
- Change the IU Tier 3 worker nodes have public IPs to avoid the head node bottleneck and retest.
- Turn the TWiki document into LaTeX document.

Backup Slides

Testing the Network (PerfSONAR)

- To get the most out of FAX, monitoring/testing network performance is essential.
 - There is a basic version of the standard network testing package PerfSONAR (the level 1 version) that can be easily installed at a Tier 3 to test network connectivity (actual data transfer rates and latencies) to full PerfSONAR installations (typically at Tier 1s/2s, network providers, and computer centers). NB: The level 1 install must run on an SL6 system.
- The level 1 package was installed on the IU Tier 3 storage server and many tests were run to MWT2 PerfSONAR servers.
 - Realizable transfer speeds were 4-8 Gbps (mem to mem).
 - One problem was found the upload speed was typically 75% of the download speed. This is still under investigation.

Full Documentation

- See the Tier 3 Evolution TWiki page for fuller details:
<https://twiki.cern.ch/twiki/bin/viewauth/AtlasComputing/Tier3EvolutionTestResults>
- I am working on a LaTeX document based on the above TWiki which runs to about 15 pages when converted to pdf.
 - The content of the LaTeX document is identical but the formatting is much better.

ROOT Input File URIs

- Local/d00/luehring/testdata/data12_8TeV.00208781...
- US ATLAS FAX server: root://glrd.usatlas.org:1094//atlas/dq2/data12_8TeV/NTUP_SMWZ/f472_m1208_p1067_p1141/data12_8TeV.00208781...
- MWT2 FAX server: root://fax.mwt2.org//atlas/dq2/data12_8TeV/NTUP_SMWZ/f495_m1266_p1328_p1329/data12_8TeV.00215091.physics_Egamma.merge.NTUP_SMWZ.f495_m1266_p1328_p1329_tid01128241_00/
 - Earlier forms of this URI are not shown.
- MWT2 FAX server using RUCIO: root://fax.mwt2.org//atlas/rucio/data12_8TeV: